Faster Evaluation of SBoxes via Common Shares

Jean-Sébastien Coron, Aurélien Greuet, Emmanuel Prouff, Rina Zeitoun

Faster Evaluation of SBoxes via Common Shares

Jean-Sébastien Coron[1], Aurélien Greuet[2], Emmanuel Prouff[3⋆], and Rina Zeitoun[2]

[1] University of Luxembourg
jean-sebastien.coron@uni.lu
[2] Oberthur Technologies, France
{r.zeitoun,a.greuet}@oberthur.com
[3] Sorbonne Universités, UPMC Univ Paris 06, CNRS, INRIA, Laboratoire d'Informatique de Paris 6 (LIP6), Équipe PolSys, 4 place Jussieu, 75252 Paris Cedex 05, France

Abstract. We describe a new technique for improving the efficiency of the masking countermeasure against side-channel attacks. Our technique is based on using common shares between secret variables, in order to reduce the number of finite field multiplications. Our algorithms are proven secure in the ISW probing model with $n \geqslant t+1$ shares against t probes. For AES, we get an equivalent of 2.8 non-linear multiplications for every SBox evaluation, instead of 4 in the Rivain-Prouff countermeasure. We obtain similar improvements for other block-ciphers. Our technique is easy to implement and performs relatively well in practice, with roughly a 20% speed-up compared to existing algorithms.

1 Introduction

Side-Channel Attacks. Side-channel analysis is a class of cryptanalytic attacks that exploit the physical environment of a cryptosystem to recover some *leakage* about its secrets. It is often more efficient than a cryptanalysis mounted in the so-called *black-box model* where no leakage occurs. In particular, *continuous side-channel attacks* in which the adversary gets information at each invocation of the cryptosystem are especially threatening. Common attacks as those exploiting the running-time, the power consumption or the electromagnetic radiations of a cryptographic computation fall into this class. Many implementations of block ciphers have been practically broken by continuous side-channel analysis and securing them has been a longstanding issue for the embedded systems industry.

The Masking Countermeasure. A sound approach to counteract side-channel attacks is to use *secret sharing* [Bla79, Sha79], often called *masking* in the context of side-channel attacks. This approach consists in splitting each sensitive

⋆ Part of this work has been done at Safran Identity and Security, and while the author was at ANSSI, France.

variable x of the implementation into n shares such that $x = x_1 \oplus \cdots \oplus x_n$, where n is called the *sharing order*, such that x can be recovered from these shares but no information can be recovered from fewer than n shares. It has been shown that the complexity of mounting a successful side-channel attack against a masked implementation increases exponentially with the order [CJRR99,PR13,DDF14]. Starting from this observation, the design of efficient *secure schemes* for different ciphers has become a foreground issue. When specified *at order* n, such a scheme aims at specifying how to update the sharing of the internal state throughout the processing while ensuring that (1) the final sharing corresponds to the expected ciphertext, and (2) the n-th order security property is satisfied.

The ISW Probing Model. Ishai, Sahai and Wagner [ISW03] initiated the theoretical study of securing circuits against an adversary who can probe a fraction of its wires. They showed how to transform any circuit of size $|C|$ into a circuit of size $\mathcal{O}(|C| \cdot t^2)$ secure against any adversary who can probe at most t wires. The ISW constructions consists in secret-sharing every variable x into $x = x_1 \oplus x_2 \oplus \cdots \oplus x_n$ where $x_2, \ldots, x_n$ are uniformly and independently distributed bits, with $n \geqslant 2t + 1$ to get security against t probes. Processing a XOR gate is straightforward as the shares can be xored separately. The processing of an AND gate $z = xy$ is based on writing:

$$z = xy = \left(\bigoplus_{i=1}^{n} x_i\right) \cdot \left(\bigoplus_{i=1}^{n} y_i\right) = \bigoplus_{1 \leqslant i,j \leqslant n} x_i y_j \tag{1}$$

where the cross-products $x_i y_j$ are first computed and then randomly recombined to get an n-sharing of the output z. This construction, called *ISW gadget* in the rest of this paper, enables, in its general form, to securely evaluate a multiplication at the cost of n^2 multiplications, $2n(n-1)$ additions and $n(n-1)/2$ random values. Its complexity is therefore $\mathcal{O}(n^2)$, which implies that the new circuit with security against t probes has $\mathcal{O}(|C| \cdot t^2)$ gates.

A proof of security in the ISW framework is usually simulation based: one must show that any set of t probes can be perfectly simulated without the knowledge of the original variables of the circuit. In [ISW03] and subsequent work this is done by progressively generating a subset I of input shares such that the knowledge of those input shares is sufficient to simulate all the t probes. For example, in the above AND gate, if the adversary would probe $x_i \cdot y_j$, one would put both indices i and j in I, so that the simulator would get the input shares x_i and y_j, and therefore could simulate the product $x_i \cdot y_j$. More generally in the ISW construction every probe adds at most two indices in I, which implies $|I| \leqslant 2t$. Therefore if the number of shares n is such that $n \geqslant 2t + 1$, then $|I| < n$, which implies that only a proper subset of the input shares is required for the simulation; those input shares can in turn be generated as independently uniformly distributed bits. Therefore, the knowledge of the original circuit variables is not required to generate a perfect simulation of the t probes, hence these probes do not bring any additional information to the attacker (since he could perform that simulation by himself).

Existing work. In the last decade, several masking countermeasures have been proposed for block-ciphers together with security proofs in the ISW probing model, based on the original notion of private circuits introduced in [ISW03]. Except [Cor14] which extends the original idea of [KJJ99] to any order, the other proposals are based on the ISW gadget recalled above. The core idea of the latter works is to split the processing into a short sequence of field multiplications and $\mathbb{F}_2$-linear operations, and then to secure these operations independently, while ensuring that the local security proofs can be combined to prove the security of the entire processing. When parametrized at order n, as recalled above the complexity of the ISW gadget for the field multiplication is $\mathcal{O}(n^2)$, but only $\mathcal{O}(n)$ for $\mathbb{F}_2$-linear operations.[1] Therefore, an interesting problem is to minimize the number of field multiplications required to evaluate an SBox.

In the Rivain-Prouff countermeasure [RP10], the authors showed how to adapt the ISW circuit construction to a software implementation of AES, by working in $\mathbb{F}_{2^8}$ instead of $\mathbb{F}_2$. Namely as illustrated in Fig. 1, the non-linear part $S(x) = x^{254}$ of the AES SBox can be evaluated with only 4 non-linear multiplications over $\mathbb{F}_{2^8}$, and a few linear squarings. Each of those 4 multiplications can in turn be evaluated with the previous ISW gadget based on Equation (1), by working over $\mathbb{F}_{2^8}$ instead of $\mathbb{F}_2$.

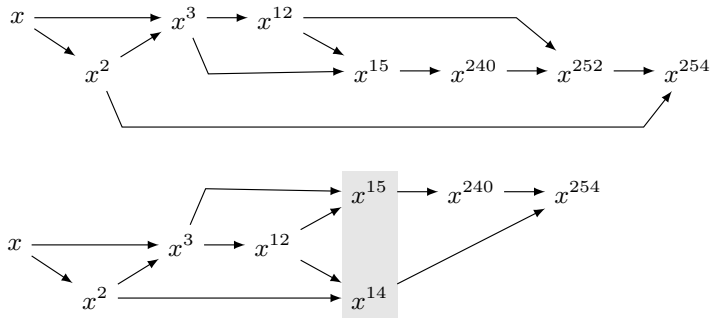

Fig. 1. a) Sequential computation of x^{254} as used in [RP10, BBD+15a]. b) Alternative computation of x^{254}; the multiplications $x^{14} = x^{12} \cdot x^2$ and $x^{15} = x^{12} \cdot x^3$ can be computed in parallel [GHS12].

The Rivain-Prouff countermeasure was later extended by Carlet *et al.* to any look-up table [CGP+12]. Namely any given k-bit SBox can be represented by a polynomial $\sum_{i=0}^{2^k-1} a_i\, x^i$ over $\mathbb{F}_{2^k}$ using Lagrange's interpolation theorem.

[1] A function f is $\mathbb{F}_2$-linear if it satisfies $f(x \oplus y) = f(x) \oplus f(y)$ for any pair (x, y) of elements in its domain. This property must not be confused with $\mathbb{F}_{2^m}$-linearity of a function, where m divides n and is larger than 1, which is defined such that $f(ax \oplus by) = af(x) \oplus bf(y)$, for every $a, b \in \mathbb{F}_{2^m}$. An $\mathbb{F}_{2^m}$-linear function is $\mathbb{F}_2$-linear but the converse is false in general.

Therefore one can mask any SBox by securely evaluating this polynomial using n-shared multiplications as in the Rivain-Prouff countermeasure. To improve efficiency, one must look for operations sequences (*e.g.* SBox representations) that minimize the number of field multiplications which are not $\mathbb{F}_2$-linear [2] (this kind of multiplication shall be called *non-linear* in this paper). This problematic has been tackled out in [CGP+12], [RV13] and [CRV14] and led to significantly reduce the number of multiplications needed to evaluate any function defined over $\mathbb{F}_{2^k}$ for $k \leqslant 10$ (*e.g.* the AES SBox can be evaluated with only 4 multiplications, and only 4 multiplications are needed for the DES SBoxes).

Recently, a sequence of works continued to improve the original work [ISW03] and led, in particular, to exhibit a new scheme enabling to securely evaluate any function of algebraic degree 2 at the cost of a single multiplication (with the ISW gadget). The application of this work to the AES SBox led the authors of [GPS14] to describe a scheme which can be secure at any order n and is a valuable alternative to the scheme proposed in [RP10]. In parallel, some schemes [BGN+14, NRS11, PR11] have been proposed which remain secure in the probing model even in presence of so-called glitches [MS06] and the recent work [RBN+15] has investigated relations between these schemes and the ISW construction.

Refined Security Model: *t*-SNI Security. Since in this paper we are interested in efficiency improvements, we would like to use the optimal $n = t + 1$ number of shares instead of $n = 2t + 1$ as in the original ISW countermeasure. For $n \geqslant 2t+1$ shares the security proof for the single ISW multiplication gadget easily extends to the full circuit [ISW03]; however for $n \geqslant t + 1$ shares only one must be extra careful. For example, for the Rivain-Prouff countermeasure, it was originally claimed in [RP10] that only $n \geqslant t + 1$ shares were required, but an attack of order $\lceil (n - 1)/2 \rceil + 1$ was later described in [CPRR13]; the security proof in [RP10] with $n \geqslant t + 1$ shares actually applies only when the ISW multiplication is used in isolation, but not for the full block-cipher.

To prove security with $n \geqslant t+1$ shares only for the full block-cipher, a refined security model against probing attacks was recently introduced in [BBD+15a], called t-SNI security. As shown in [BBD+15a], this stronger definition of t-SNI security enables to prove that a gadget can be used in a full construction with $n \geqslant t + 1$ shares, instead of $n \geqslant 2t + 1$ for the weaker definition of t-NI security (corresponding to the original ISW security proof). The authors show that the ISW multiplication gadget does satisfy this stronger t-SNI security definition. They also show that with some additional mask refreshing, the Rivain-Prouff countermeasure for the full AES can be made secure with $n \geqslant t + 1$ shares. Due to its power and simplicity, the t-SNI notion appears to be the "right" security definition against probing attacks. Therefore, in this paper, we always prove the security of our algorithms under this stronger t-SNI notion, so that our

[2] A multiplication over a field of characteristic 2 is $\mathbb{F}_2$-linear if it corresponds to a Frobenius automorphism, *i.e.* to a series of squarings.

algorithms can be used within a larger construction (typically a full block-cipher) with $n \geqslant t+1$ shares only.

Our Contribution. Our goal in this paper is to further improve the efficiency of the masking countermeasure. As recalled above, until now the strategy followed by the community has been to reduce the number of calls to the ISW multiplication gadget. In this paper, we follow a complementary approach consisting in reducing the complexity of the ISW multiplication gadget itself. Our core idea is to use common shares between the inputs of multiple ISW multiplication gadgets, up to the first $n/2$ shares; in that case, a given processing performed in the first ISW gadget can be re-used in subsequent gadgets.

Consider for example the alternative evaluation circuit for x^{254} in AES used in [GHS12], as illustrated in Fig. 1. It still has 4 non-linear multiplications as in the original circuit [RP10], but now the two multiplications $x^{14} \leftarrow x^{12} \cdot x^2$ and $x^{15} \leftarrow x^{12} \cdot x^3$ can be evaluated in parallel, moreover with a common operand x^{12}. Let denote by $d \leftarrow c \cdot a$ and $e \leftarrow c \cdot b$ those two multiplications with common operand c. In the ISW multiplication gadget, one must compute all cross-products $c_i \cdot a_j$ and $c_i \cdot b_j$ for all $1 \leqslant i, j \leqslant n$. Now if we can ensure that half of the shares of a and b are the same, that is $a_j = b_j$ for all $1 \leqslant j \leqslant n/2$, then the products $c_i \cdot a_j$ and $c_i \cdot b_j$ for $1 \leqslant j \leqslant n/2$ are the same and can be computed only once; see Fig. 2 for an illustration. This implies that when processing the second multiplication gadget for $e \leftarrow c \cdot b$, we only have to compute $n^2/2$ finite field multiplications instead of n^2. For two multiplications as above, this saves the equivalent of 0.5 multiplication.

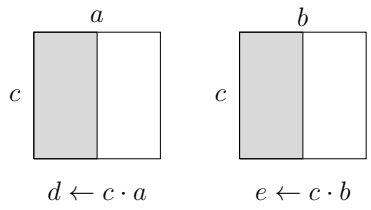

Fig. 2. When half of the shares in a and b are the same, the multiplications corresponding to the left-hand blocks are the same. This saves the equivalent of 0.5 multiplications out of 2.

To ensure that the two inputs have half of their shares in common, we introduce a new gadget called CommonShares with complexity $\mathcal{O}(n)$, taking as input two independent n-sharings of data and outputting two new n-sharings, but with their first $n/2$ shares in common. Obviously this must be achieved without degrading the security level; we show that this is indeed the case by proving the security of the full SBox evaluation in the previous t-SNI model, with $n \geqslant t+1$ shares. Note that we cannot have more than $n/2$ shares in common between two variables a and b, since otherwise there would be a straightforward attack with fewer than n probes: namely if $a_i = b_i$ for all $1 \leqslant i \leqslant k$, then we can probe the

$2(n-k)$ remaining shares a_i and b_i for $k+1 \leqslant i \leqslant n$; if $k > n/2$ this gives strictly less than n shares, whose xor gives the secret variable $a \oplus b$. Hence having half of the shares in common is optimal.

More generally, the 16 SBoxes of AES can be processed in parallel, and therefore each of the 4 non-linear multiplications in x^{254} can be processed in parallel. As opposed to the previous case those multiplications do not share any operand, but we show that by using a generalization of the CommonShares algorithm between m operands instead of 2, for every multiplication in the original circuit one can still save the equivalent of roughly $1/4$ multiplication. This also applies to other block-ciphers as well, since in most block-ciphers the SBoxes are applied in parallel. One can therefore apply the technique from [CRV14] based on fast polynomial evaluation, and using our CommonShares algorithm between the inputs of the evaluated polynomials, we again save roughly $1/4$ of the number of finite field multiplications. Our results for various block-ciphers are summarized in Table 1, in which we give the equivalent number of non-linear multiplications for a single SBox evaluation, for various block-ciphers; we refer to Section 5 for a detailed description. Finally, we show in the full version of this paper [CGPZ16] how to apply our common shares technique to the Threshold Implementations (TI) approach for securing implementation against side channel attacks, even in the presence of glitches.

	SBox					
Methods	AES	DES	PRESENT	SERPENT	CAMELLIA	CLEFIA
Parity-Split [CGP+12]	4	10	3	3	22	22
Roy-Vivek [RV13]	4	7	3	3	15	15,16
[CRV14]	4	4	2	2	10	10
Our Method	**2.8**	**3.1**	**1.5**	**1.5**	**7.8**	**7.8**

Table 1. Equivalent number of non-linear multiplications for a single SBox evaluation, for various block-ciphers.

Practical Implementation. A practical implementation of our common shares technique is described in Section 7, for the n-shared evaluation of x^{254} in AES, on ATmega1284P (8-bit AVR microcontroller) and ARM Cortex M0 (32-bit CPU). We obtain that our technique is relatively practical: for a large number of shares, we get roughly a 20% speed improvement compared to the Rivain-Prouff countermeasure (but only roughly 5% compared to the quadratic evaluation technique in [GPS14]).

2 Security Definitions

Given a variable $x \in \mathbb{F}_{2^k}$ and an integer n, we say that the vector $(x_1, \ldots, x_n) \in (\mathbb{F}_{2^k})^n$ is an n-sharing of x if $x = \bigoplus_{i=1}^{n} x_i$. We recall the security definitions from [BBD+15a], which we make slightly more explicit. For simplicity we only

provide the definitions for a simple gadget taking as input a single variable x (given by n shares x_i) and outputting a single variable y (given by n shares y_i). We provide the generalization to multiple inputs and outputs in the full version of this paper [CGPZ16]. Given a vector $(x_i)_{1 \leqslant i \leqslant n}$, we denote by $x_{|I} := (x_i)_{i \in I}$ the sub-vector of shares x_i with $i \in I$.

Definition 1 (t-NI security). *Let G be a gadget taking as input $(x_i)_{1 \leqslant i \leqslant n}$ and outputting $(y_i)_{1 \leqslant i \leqslant n}$. The gadget G is said t-NI secure if for any set of t_1 intermediate variables and any subset $\mathcal{O}$ of output indices, there exists a subset I of input indices with $|I| \leqslant t_1 + |\mathcal{O}|$, such that the t_1 intermediate variables and the output variables $y_{|\mathcal{O}}$ can be perfectly simulated from $x_{|I}$.*

Definition 2 (t-SNI security). *Let G be a gadget taking as input $(x_i)_{1 \leqslant i \leqslant n}$ and outputting $(y_i)_{1 \leqslant i \leqslant n}$. The gadget G is said t-SNI secure if for any set of t_1 intermediate variables and any subset $\mathcal{O}$ of output indices such that $t_1 + |\mathcal{O}| \leqslant t$, there exists a subset I of input indices with $|I| \leqslant t_1$, such that the t_1 intermediate variables and the output variables $y_{|\mathcal{O}}$ can be perfectly simulated from $x_{|I}$.*

The t-NI security notion corresponds to the original security definition in the ISW probing model; it allows to prove the security of a full construction with $n \geqslant 2t + 1$ shares. The stronger t-SNI notion allows to prove the security of a full construction with $n \geqslant t + 1$ shares only [BBD+15a]. The difference is that in the stronger t-SNI notion, the size of the input shares subset I can only depend on the number of internal probes t_1, and must be independent of the number of output variables $|\mathcal{O}|$ that must be simulated (as long as the condition $t_1 + |\mathcal{O}| \leqslant t$ is satisfied). Intuitively, this provides an "isolation" between the output shares and the input shares of a given gadget, and for composed constructions this enables to easily prove that a full construction is t-SNI secure, based on the t-SNI security of its components.

3 The Rivain-Prouff Countermeasure

In this section we recall the Rivain-Prouff countermeasure [RP10] for securing AES against high-order attacks. It can be seen as an extension to $\mathbb{F}_{2^k}$ of the original ISW countermeasure [ISW03] described in $\mathbb{F}_2$. The Rivain-Prouff countermeasure is proved t-SNI secure in [BBD+15a]; therefore it can be used to protect a full block-cipher against t probes with $n \geqslant t + 1$ shares, instead of $n \geqslant 2t + 1$ shares in the original ISW probing model.

3.1 The Rivain-Prouff Multiplication

The Rivain-Prouff countermeasure is based on the SecMult operation below, which is similar to the ISW multiplication gadget but over $\mathbb{F}_{2^k}$ instead of $\mathbb{F}_2$. The SecMult algorithm enables to securely compute a product $c = a \cdot b$ over $\mathbb{F}_{2^k}$, from an n-sharing of a and b, and outputs an n-sharing of c. Here we use the linear memory version from [Cor14], using similar notations as in [BBD+15a].

Algorithm 1 SecMult

Require: shares a_i satisfying $\bigoplus_{i=1}^{n} a_i = a$, shares b_i satisfying $\bigoplus_{i=1}^{n} b_i = b$
Ensure: shares c_i satisfying $\bigoplus_{i=1}^{n} c_i = a \cdot b$

1: **for** $i = 1$ **to** n **do**
2: $\quad c_i \leftarrow a_i \cdot b_i$
3: **end for**
4: **for** $i = 1$ **to** n **do**
5: $\quad$ **for** $j = i + 1$ **to** n **do**
6: $\quad\quad r \leftarrow \mathbb{F}_{2^k}$ ▷ referred by $r_{i,j}$
7: $\quad\quad c_i \leftarrow c_i \oplus r$ ▷ referred by $c_{i,j}$
8: $\quad\quad r \leftarrow (a_i \cdot b_j \oplus r) \oplus a_j \cdot b_i$ ▷ referred by $r_{j,i}$
9: $\quad\quad c_j \leftarrow c_j \oplus r$ ▷ referred by $c_{j,i}$
10: $\quad$ **end for**
11: **end for**
12: **return** $(c_1, \ldots, c_n)$

It is shown in [BBD+15a] that the SecMult algorithm is t-SNI secure with $n \geqslant t+1$ shares. For completeness we provide a proof of Lemma 1 in the full version of this paper [CGPZ16]; our proof is essentially the same as in [BBD+15a]. In the full version of this paper [CGPZ16], we also provide a slightly different, more modular proof in which we separate the computation of the matrix elements $v_{ij} = a_i \cdot b_j$ from the derivation of the output shares c_i.

Lemma 1 (t-SNI of SecMult). *Let $(a_i)_{1 \leqslant i \leqslant n}$ and $(b_i)_{1 \leqslant i \leqslant n}$ be the input shares of the SecMult operation, and let $(c_i)_{1 \leqslant i < n}$ be the output shares. For any set of t_1 intermediate variables and any subset $|\mathcal{O}| \leqslant t_2$ of output shares such that $t_1 + t_2 < n$, there exist two subsets I and J of indices with $|I| \leqslant t_1$ and $|J| \leqslant t_1$, such that those t_1 intermediate variables as well as the output shares $c_{|\mathcal{O}}$ can be perfectly simulated from $a_{|I}$ and $b_{|J}$.*

3.2 Mask Refreshings

To obtain security against t probes with $n \geqslant t+1$ shares instead of $n \geqslant 2t+1$, the previous SecMult algorithm is usually not sufficient; one must also use a mask refreshing algorithm. The following RefreshMask operation is used in [BBD+15a] to get the t-SNI security of a full construction.

The following lemma is proven in [BBD+15a], showing the t-SNI security of RefreshMask. In the full version of this paper [CGPZ16] we also provide a modular proof, using the same approach as in Lemma 1; namely the above RefreshMask algorithm can be viewed as a SecMult with multiplication by 1, with shares $(1, 0, \ldots, 0)$; therefore the same proof technique applies.

Lemma 2 (t-SNI of RefreshMask). *Let $(a_i)_{1 \leqslant i \leqslant n}$ be the input shares of the RefreshMask operation, and let $(c_i)_{1 \leqslant i \leqslant n}$ be the output shares. For any set of t_1 intermediate variables and any subset $|\mathcal{O}| \leqslant t_2$ of output shares such that $t_1 + t_2 < n$, there exists a subset I of indices with $|I| \leqslant t_1$, such that the t_1*

Algorithm 2 RefreshMask

Input: $a_1, \ldots, a_n$
Output: $c_1, \ldots, c_n$ such that $\bigoplus_{i=1}^n c_i = \bigoplus_{i=1}^n a_i$

For $i = 1$ **to** n **do** $c_i \leftarrow a_i$
for $i = 1$ to n **do**
  for $j = i + 1$ to n **do**
    $r \leftarrow \{0,1\}^k$
    $c_i \leftarrow c_i \oplus r$
    $c_j \leftarrow c_j \oplus r$
  end for
end for
return $c_1, \ldots, c_n$

intermediate variables as well as the output shares $c_{|\mathcal{O}}$ can be perfectly simulated from $a_{|I}$.

3.3 Application to the Computation of x^{254} in $\mathbb{F}_{2^8}$

To compute $y = x^{254}$ over $\mathbb{F}_{2^8}$ with 4 multiplications, the following sequence of operation is used in [RP10], including two RefreshMask operations.

Algorithm 3 SecExp254

Input: shares $x_1, \ldots, x_n$ satisfying $x = \bigoplus_{i=1}^n x_i$
Output: shares $y_1, \ldots, y_n$ such that $\bigoplus_{i=1}^n y_i = x^{254}$

For $i = 1$ to n **do** $z_i \leftarrow x_i^2$ ▷ $\bigoplus_i z_i = x^2$
$(z_i)_{1 \leqslant i \leqslant n} \leftarrow \mathsf{RefreshMask}((z_i)_{1 \leqslant i \leqslant n})$
$(y_i)_{1 \leqslant i \leqslant n} \leftarrow \mathsf{SecMult}((z_i)_{1 \leqslant i \leqslant n}, (x_i)_{1 \leqslant i \leqslant n})$ ▷ $\bigoplus_i y_i = x^3$
For $i = 1$ to n **do** $w_i \leftarrow y_i^4$ ▷ $\bigoplus_i w_i = x^{12}$
$(w_i)_{1 \leqslant i \leqslant n} \leftarrow \mathsf{RefreshMask}((w_i)_{1 \leqslant i \leqslant n})$
$(y_i)_{1 \leqslant i \leqslant n} \leftarrow \mathsf{SecMult}((y_i)_{1 \leqslant i \leqslant n}, (w_i)_{1 \leqslant i \leqslant n})$ ▷ $\bigoplus_i y_i = x^{15}$
For $i = 1$ to n **do** $y_i \leftarrow y_i^{16}$ ▷ $\bigoplus_i y_i = x^{240}$
$(y_i)_{1 \leqslant i \leqslant n} \leftarrow \mathsf{SecMult}((y_i)_{1 \leqslant i \leqslant n}, (w_i)_{1 \leqslant i \leqslant n})$ ▷ $\bigoplus_i y_i = x^{252}$
$(y_i)_{1 \leqslant i \leqslant n} \leftarrow \mathsf{SecMult}((y_i)_{1 \leqslant i \leqslant n}, (z_i)_{1 \leqslant i \leqslant n})$ ▷ $\bigoplus_i y_i = x^{254}$
return $y_1, \ldots, y_n$

Using the two previous lemmas, one can prove the t-SNI security of SecExp254; we refer to [BBD+15a] for the proof.

Lemma 3 (t-SNI of x^{254}). *Let $(x_i)_{1 \leqslant i \leqslant n}$ be the input shares of* ExpSec254*, and let $(y_i)_{1 \leqslant i \leqslant n}$ be the output shares. For any set of t_1 intermediate variables and any subset $|\mathcal{O}| \leqslant t_2$ of output shares such that $t_1 + t_2 < n$, there exists a subset I of indices with $|I| \leqslant t_1$, such that those t_1 intermediate variables as well as the output shares $y_{|\mathcal{O}}$ can be perfectly simulated from $x_{|I}$.*

As explained in [BBD+15a], since the SecExp254 operation has the t-SNI property, it can be used to secure a full AES against t probes with $n \geqslant t+1$ shares.

4 Secure Computation of 2 Parallel Multiplications with Common Operand, and Application to AES

In this section we show a first efficiency improvement of the Rivain-Prouff countermeasure for AES recalled in the previous section. Namely, we show that when two finite-field multiplications $d \leftarrow c \cdot a$ and $e \leftarrow c \cdot b$ have the same operand c, we can save $n^2/2$ field multiplications in SecMult by making sure that the inputs a and b have half of their shares in common; we then show how to apply this technique to the evaluation of the AES SBox, by using an alternative evaluation circuit for x^{254}.

Arithmetic Circuit with Depth 3 for x^{254}. The original arithmetic circuit for computing $y = x^{254}$ over $\mathbb{F}_{2^8}$ from [RP10] and recalled in Section 3.3 has 4 multiplicative levels, with a total of 4 non-linear multiplications. Below we use an alternative circuit with only 3 multiplicative levels, still with 4 multiplications, as described in [GHS12]; see Fig. 1 for an illustration.

- Level 1: compute $x^3 = x \cdot x^2$ (1 mult) and then $x^{12} = (x^3)^4$.
- Level 2: compute $x^{14} = x^{12} \cdot x^2$ (1 mult) and $x^{15} = x^{12} \cdot x^3$ (1 mult), and then $x^{240} = (x^{15})^{16}$.
- Level 3: compute $x^{254} = x^{240} \cdot x^{14}$ (1 mult).

Multiplications with Common Shares. In the arithmetic circuit above, the multiplications $x^{14} \leftarrow x^{12} \cdot x^2$ and $x^{15} \leftarrow x^{12} \cdot x^3$ can be computed in parallel; moreover they have one operand x^{12} in common. More generally, assume that we must compute two multiplications with a common operand c:

$$d \leftarrow c \cdot a$$
$$e \leftarrow c \cdot b$$

The SecMult algorithm will compute the cross-products $c_i \cdot a_j$ and $c_i \cdot b_j$ for all $1 \leqslant i, j \leqslant n$. Now assume that half of the shares of a and b are the same, that is $a_j = b_j$ for all $1 \leqslant j \leqslant n/2$. In that case the products $c_i \cdot a_j$ for $1 \leqslant j \leqslant n/2$ have to be computed only once, and therefore when processing $e \leftarrow c \cdot b$, we only have to compute $n^2/2$ multiplications instead of n^2; see Fig. 2 for an illustration. For an arithmetic circuit with 4 multiplications as above, this saves the equivalent of 0.5 multiplication.

4.1 The CommonShares Algorithm

The CommonShares algorithm below ensures that the output shares a'_i and b'_i corresponding to a and b are the same on the first half, that is $a'_i = b'_i$ for all $1 \leqslant i \leqslant n/2$. In the rest of the paper, for simplicity we assume that n is even.

Algorithm 4 CommonShares

Require: shares a_i satisfying $\bigoplus_{i=1}^{n} a_i = a$, shares b_i satisfying $\bigoplus_{i=1}^{n} b_i = b$
Ensure: shares a'_i and b'_i satisfying $\bigoplus_{i=1}^{n} a'_i = a$ and $\bigoplus_{i=1}^{n} b'_i = b$, with $a'_i = b'_i$ for all $1 \leqslant i \leqslant n/2$

1: **for** $i = 1$ **to** $n/2$ **do**
2: $\quad r_i \leftarrow^{\$} \mathbb{F}_{2^k}$
3: $\quad a'_i \leftarrow r_i,\ \ a'_{n/2+i} \leftarrow (a_{n/2+i} \oplus r_i) \oplus a_i$ $\quad \triangleright\ a'_i \oplus a'_{n/2+i} = a_i \oplus a_{n/2+i}$
4: $\quad b'_i \leftarrow r_i,\ \ b'_{n/2+i} \leftarrow (b_{n/2+i} \oplus r_i) \oplus b_i$ $\quad \triangleright\ b'_i \oplus b'_{n/2+i} = b_i \oplus b_{n/2+i}$
5: **end for**
6: **return** $(a'_i)_{1 \leqslant i \leqslant n}$ and $(b'_i)_{1 \leqslant i \leqslant n}$

It is easy to see that we still get as output an n-sharing of the same variables a and b, since for each $1 \leqslant i \leqslant n/2$ we have $a'_i \oplus a'_{n/2+i} = a_i \oplus a_{n/2+i}$, and similarly for b. As explained previously, we cannot have more than $n/2$ shares in common between a and b, since otherwise there would be a straightforward attack with fewer than n probes: namely if $a_i = b_i$ for all $1 \leqslant i \leqslant k$, then we can probe the $2(n-k)$ remaining shares a_i and b_i for $k+1 \leqslant i \leqslant n$; if $k > n/2$ this gives strictly less than n shares, whose xor gives the secret variable $a \oplus b$. Hence having half of the shares in common is optimal.

The following Lemma shows the security of the CommonShares algorithm; as will be shown later, for this algorithm we only need the weaker t-NI security property (instead of t-SNI).

Lemma 4 (t-NI of CommonShares). *Let $(a_i)_{1 \leqslant i \leqslant n}$ and $(b_i)_{1 \leqslant i \leqslant n}$ be the input shares of the algorithm* CommonShares*, and let $(a'_i)_{1 \leqslant i \leqslant n}$ and $(b'_i)_{1 \leqslant i \leqslant n}$ be the output shares. For any set of t_1 intermediate variables and any subsets of indices $I, J \subset [1, n]$, there exists a subset $\mathcal{S} \subset [1, n]$ with $|\mathcal{S}| \leqslant |I| + |J| + t_1$, such that those t_1 variables as well as the output shares $a'_{|I}$ and $b'_{|J}$ can be perfectly simulated from $a_{|\mathcal{S}}$ and $b_{|\mathcal{S}}$.*

Proof. The proof intuition is as follows. If for a given i with $1 \leqslant i \leqslant n/2$ the adversary requests only one of the variables r_i, $a_{n/2+i} \oplus r_i$, $b_{n/2+i} \oplus r_i$, $a'_{n/2+i}$ or $b'_{n/2+i}$, then such variable can be perfectly simulated without knowing any of the input shares a_i, b_i, $a_{n/2+i}$ and $b_{n/2+i}$, thanks to the mask r_i. On the other hand, if two such variables (or more) are requested, then we can provide a perfect simulation from the 4 previous input shares, whose knowledge is obtained by adding the two indices i and $n/2 + i$ in $\mathcal{S}$. Therefore we never add more than one index in $\mathcal{S}$ per probe (or per output index in I or J), which implies that the size of the subset $\mathcal{S}$ of input shares is upper-bounded by $|I| + |J| + t_1$, as required.[3]

More precisely, we describe hereafter the construction of the set $\mathcal{S} \subset [1, n]$ of input shares, initially empty. For every probed input variable a_i and b_i (for

[3] Note that the proof would not work without the masks r_i; namely with $r_i = 0$ we would need to know both a_i and $a_{n/2+i}$ to simulate $a'_{n/2+i}$; hence with t probes we would need at least $n \geqslant 2t + 1$ shares, which would make CommonShares useless.

any i), we add i to $\mathcal{S}$. For all $1 \leqslant i \leqslant n/2$, we let t_i be the number of probed variables among $a_{n/2+i} \oplus r_i$ and $b_{n/2+i} \oplus r_i$. We let:

$$\lambda_i := t_i \; + \; |\{i, n/2+i\} \cap I| \; + \; |\{i, n/2+i\} \cap J| \; ,$$

We then add $\{i, n/2+i\}$ to $\mathcal{S}$ if $\lambda_i \geqslant 2$. This terminates the construction of $\mathcal{S}$. By construction of $\mathcal{S}$, we must have $|\mathcal{S}| \leqslant |I| + |J| + t$ as required.

We now show that the output shares $a'_{|I}$ and $b'_{|J}$ and the t_1 intermediate variables of Algorithm CommonShares can be perfectly simulated from $a_{|\mathcal{S}}$ and $b_{|\mathcal{S}}$. This is clear for the probed input variables a_i and b_i. For all $1 \leqslant i \leqslant n/2$, we distinguish two cases. If $\lambda_i \geqslant 2$, then $\{i, n/2+i\} \in \mathcal{S}$, so we can let $r_i \leftarrow \mathbb{F}_{2^k}$ as in the real algorithm and simulate all output and intermediate variables from the knowledge of a_i, $a_{n/2+i}$, b_i and $b_{n/2+i}$. If $\lambda_i = 1$, then if $t_i = 0$, then only a single output variable among a'_i, b'_i, $a'_{n/2+i}$ and $b'_{n/2+i}$ must be simulated. Since each of those variables is masked by r_i, we can simulate this single output variable by generating a random value in $\mathbb{F}_{2^k}$. Similarly, if $t_i = 1$, then only one of the two intermediate variables among $a_{n/2+i} \oplus r_i$ and $b_{n/2+i} \oplus r_i$ is probed (while no output variable must be simulated), and therefore we can also simulate such variable by generating a random value in $\mathbb{F}_{2^k}$. This terminates the proof of Lemma 4. □

4.2 The CommonMult Algorithm

To perform the two multiplications with the same operand $d \leftarrow c \cdot a$ and $e \leftarrow c \cdot b$, instead of doing two independent SecMult, we define the following CommonMult algorithm below.

Algorithm 5 CommonMult

Input: shares satisfying $c = \bigoplus_{i=1}^{n} c_i$, $a = \bigoplus_{i=1}^{n} a_i$ and $b = \bigoplus_{i=1}^{n} b_i$.
Output: d_i such that $\bigoplus_{i=1}^{n} d_i = c \cdot a$, and e_i such that $\bigoplus_{i=1}^{n} e_i = c \cdot b$
1: $(a'_i)_{1 \leqslant i \leqslant n}, (b'_i)_{1 \leqslant i \leqslant n} \leftarrow \mathsf{CommonShares}((a_i)_{1 \leqslant i \leqslant n}, (b_i)_{1 \leqslant i \leqslant n})$
2: $(d_i)_{1 \leqslant i \leqslant n} \leftarrow \mathsf{SecMult}((c_i)_{1 \leqslant i \leqslant n}, (a'_i)_{1 \leqslant i \leqslant n})$
3: $(e_i)_{1 \leqslant i \leqslant n} \leftarrow \mathsf{SecMult}((c_i)_{1 \leqslant i \leqslant n}, (b'_i)_{1 \leqslant i \leqslant n})$
4: **return** $(d_i)_{1 \leqslant i \leqslant n}$ and $(e_i)_{1 \leqslant i \leqslant n}$.

The algorithm first calls the previous CommonShares subroutine, to ensure that half of the shares of a and b are the same. It then applies the previous SecMult algorithm twice to securely compute the two multiplications. Then the multiplications $c_i \cdot a_j$ for $1 \leqslant j \leqslant n/2$ performed in the first SecMult can be reused in the second SecMult, so this saves $n^2/2$ multiplications. More precisely, for the SecMult computation performed at Line 3, we don't have to compute again the products $c_i \cdot b'_j$ for $1 \leqslant j \leqslant n/2$, since those products have already been computed at Line 2 with $c_i \cdot a'_j$, since $a'_j = b'_j$ for all $1 \leqslant j \leqslant n/2$. However reusing at Line 3 the products already computed at Line 2 requires to store

$\mathcal{O}(n^2)$ values. In the full version of this paper [CGPZ16] we describe a different version of the CommonMult algorithm above, where the matrix elements $c_i \cdot a_j$ are computed on the fly and then used in both SecMult, with memory complexity $\mathcal{O}(n)$ instead of $\mathcal{O}(n^2)$.

The following Lemma shows that the CommonMult algorithm is t-SNI secure in the ISW model, with $n \geqslant t+1$ shares. We provide the proof in the full version of this paper [CGPZ16].

Lemma 5 (t-SNI of CommonMult). *Let $(a_i)_{1\leqslant i\leqslant n}$, $(b_i)_{1\leqslant i\leqslant n}$ and $(c_i)_{1\leqslant i\leqslant n}$ be the input shares of the CommonMult operation, and let $(d_i)_{1\leqslant i\leqslant n}$ and $(e_i)_{1\leqslant i\leqslant n}$ be the output shares. For any set of t_1 intermediate variables and any subsets $|\mathcal{O}_1| \leqslant t_2$ and $|\mathcal{O}_2| \leqslant t_2$ of output shares such that $t_1 + t_2 < n$, there exist two subsets I and J of indices such that $|I| \leqslant t_1$ and $|J| \leqslant t_1$, and those t_1 intermediate variables as well as the output shares $d_{|\mathcal{O}_1}$ and $e_{|\mathcal{O}_2}$ can be perfectly simulated from $a_{|J}$, $b_{|J}$ and $c_{|I}$.*

4.3 Application to AES SBoxes

We are now ready to describe the full computation of $y = x^{254}$ based on the CommonShares algorithm; the algorithm SecExp254' is described below; it is a variant of Algorithm 3.

Algorithm 6 SecExp254'

Input: shares $x_1, \ldots, x_n$ satisfying $x = \bigoplus_{i=1}^n x_i$
Output: shares $y_1, \ldots, y_n$ such that $\bigoplus_{i=1}^n y_i = x^{254}$

1: **For** $i = 1$ to n **do** $z_i \leftarrow x_i^2$ ▷ $\bigoplus_i z_i = x^2$
2: $(x_i)_{1\leqslant i\leqslant n} \leftarrow \mathsf{RefreshMask}((x_i)_{1\leqslant i\leqslant n})$
3: $(y_i)_{1\leqslant i\leqslant n} \leftarrow \mathsf{SecMult}((z_i)_{1\leqslant i\leqslant n}, (x_i)_{1\leqslant i\leqslant n})$ ▷ $\bigoplus_i y_i = x^3$
4: **For** $i = 1$ to n **do** $w_i \leftarrow y_i^4$ ▷ $\bigoplus_i w_i = x^{12}$
5: $(w_i)_{1\leqslant i\leqslant n} \leftarrow \mathsf{RefreshMask}((w_i)_{1\leqslant i\leqslant n})$
6: $(z_i)_{1\leqslant i\leqslant n}, (y_i)_{1\leqslant i\leqslant n} \leftarrow \mathsf{CommonMult}((w_i)_{1\leqslant i\leqslant n}, (z_i)_{1\leqslant i\leqslant n}, (y_i)_{1\leqslant i\leqslant n})$ ▷ $\bigoplus_i z_i = x^{14}$, $\bigoplus_i y_i = x^{15}$
7: **For** $i = 1$ to n **do** $y_i \leftarrow y_i^{16}$ ▷ $\bigoplus_i y_i = x^{240}$
8: $(y_i)_{1\leqslant i\leqslant n} \leftarrow \mathsf{SecMult}((y_i)_{1\leqslant i\leqslant n}, (z_i)_{1\leqslant i\leqslant n})$ ▷ $\bigoplus_i y_i = x^{254}$
9: **return** $y_1, \ldots, y_n$

The following Lemma proves the t-SNI security of our new algorithm; therefore our new algorithm achieves exactly the same security level as Algorithm 3. That is, it can be used in the computation of a full block-cipher, with $n \geqslant t+1$ shares against t probes. We provide the proof in the full version of this paper [CGPZ16].

Lemma 6 (t-SNI of x^{254}). *Let $(x_i)_{1\leqslant i\leqslant n}$ be the input shares of the x^{254} operation, and let $(y_i)_{1\leqslant i\leqslant n}$ be the output shares. For any set of t_1 intermediate variables and any subset $|\mathcal{O}| \leqslant t_2$ of output shares such that $t_1 + t_2 < n$, there*

exists a subset I of indices with $|I| \leqslant t_1$, such that those t_1 intermediate variables as well as the output shares $y_{|\mathcal{O}}$ can be perfectly simulated from $x_{|I}$.

Finally, we summarize in Table 2 the complexities of the above algorithms. Table 2 shows that our new algorithm for x^{254} saves $n^2/2$ multiplications, with the same security level as in the original algorithm.

	# add	# mult	# rand
SecMult (Alg. 1)	$2n^2$	n^2	$n^2/2$
RefreshMask (Alg. 2)	n^2	-	$n^2/2$
SecMult $\times$ 2	$4n^2$	$2n^2$	n^2
CommonMult (Alg. 5)	$4n^2$	$3n^2/2$	n^2
SecExp254 (Alg. 3)	$10n^2$	$4n^2$	$3n^2$
SecExp254' (Alg. 6)	$10n^2$	$7n^2/2$	$3n^2$

Table 2. Complexity of CommonMult and SecExp254'; for simplicity we omit the $\mathcal{O}(n)$ terms.

5 Parallel Multiplications with Common Shares

In the previous section, we have shown that by using a different arithmetic circuit for x^{254}, two multiplications in $\mathbb{F}_{2^8}$ could be processed in parallel, moreover with a common operand, and then by using half common shares we could save the equivalent of 1/2 multiplication out of 4 in the evaluation of an AES SBox.

In the full version of this paper [CGPZ16], we consider the case of parallel multiplications that do not necessarily share an operand. Previously we have focused on a single evaluation of an AES SBox, but in AES the 16 SBoxes can actually be processed in parallel, and therefore each of the 4 multiplications in x^{254} can be processed in parallel. As opposed to the previous case those multiplications do not share any operand, but we show that by using a generalization of the CommonShares algorithm between m operands instead of 2, for every multiplication one can still save the equivalent of roughly 1/4 multiplication.

6 Parallel Computation of Quadratic Functions

In [CPRR15], the authors propose a generalization of an idea originally published in [CPRR13] to securely process any function h of algebraic degree[4] 2, with

[4] The *algebraic degree* of a function h is the integer value $\max_{a_i \neq 0}(\mathrm{HW}(i))$ where the a_i's are the coefficients of the polynomial representation of h and where $\mathrm{HW}(i)$ denotes the Hamming weight of i.

application to the secure evaluation of SBoxes. The algorithm is based on the following equation:

$$h\left(\sum_{i=1}^{n} x_i\right) = \sum_{1\leqslant i<j\leqslant n} \big(h(x_i + x_j + s_{ij}) + h(x_i + s_{ij}) + h(x_j + s_{ij}) + h(s_{ij})\big) + \sum_{i=1}^{n} h(x_i) + ((n+1) \bmod 2) \cdot h(0) \quad (2)$$

which holds for any $s_{ij} \in \mathbb{F}_{2^k}$. From the above equation, any function h of algebraic degree 2 can be securely processed with n-th order security.

In the full version of this paper [CGPZ16], we recall the algorithm from [CPRR15] for the secure evaluation of the quadratic function $h(x)$, and its application to AES. We then show how to use our common shares technique for we provide for m parallel evaluations of $h(x)$.

7 Implementation

We have done a practical implementation of our algorithms for the AES SBox. More precisely we have implemented the n-shared evaluation of x^{254} in four different ways:

- RP10: using the Rivain-Prouff algorithm, as described in Alg. 3;
- CM: using our common shares technique, as described in Alg. 6;
- GPS14: using quadratic functions, as described in the full version of this paper [CGPZ16];
- GPS14CS: using quadratic functions and common shares, as explained in the full version of this paper [CGPZ16];

	8 shares				16 shares			
	RP10	CM	GPS14	GPS14CS	RP10	CM	GPS14	GPS14CS
ATmega	20360	18244	11076	12447	70966	57644	39554	40086
ARM	20333	18156	13796	13156	77264	65556	54133	50560

	32 shares				Ratio for 8,16 and 32 shares	
	RP10	CM	GPS14	GPS14CS	CM/RP10	GPS14CS/GPS14
ATmega	268.10^3	209.10^3	152.10^3	147.10^3	0.9, 0.81, 0.78	1.1, 1, 0.97
ARM	303.10^3	251.10^3	215.10^3	200.10^3	0.89, 0.85, 0.83	0.95, 0.93, 0.93

Table 3. Performances comparison of the RP10, CM, GPS14 and GPS14CS algorithms, on the ATmega and ARM platforms.

For portability, the code is written in `C`, except the field multiplication in $\mathbb{F}_{2^8}$ which is written in assembly for ATmega1284P (8-bit AVR microcontroller) and

for ARM Cortex M0 (32-bit CPU). Performance is evaluated using simulators (AVR Studio for ATmega, Keil uVision for ARM). We assume that the random generation of one byte takes 1 cycle. This assumption is reasonable: there are at least several dozens of cycles between two 1-byte random number requests; on chips embedding hardware RNG, this is often enough to get a random value by a single memory access, without waiting. We give the average number of cycles to compute one AES SBox among 16 SBoxes in Table 3.

Those implementation results show that our common shares technique is relatively practical: for a large number of shares, we get roughly a 20% speed improvement compared to the Rivain-Prouff countermeasure (but only roughly 5% compared to the quadratic evaluation technique in [GPS14]).

Acknowledgments. We wish to thank Sonia Belaïd who applied the EasyCrypt verification tool [BBD+15b] on our AES SBox algorithm with common shares, at order $n = 6$.

References

[BBD+15a] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, and Benjamin Grégoire. Compositional verification of higher-order masking: Application to a verifying masking compiler. Cryptology ePrint Archive, Report 2015/506, 2015. `http://eprint.iacr.org/`.

[BBD+15b] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. Verified proofs of higher-order masking. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 457–485, 2015.

[BGN+14] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-order threshold implementations. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014.

[Bla79] G.R. Blakely. Safeguarding cryptographic keys. In *National Comp. Conf.*, volume 48, pages 313–317, New York, June 1979. AFIPS Press.

[CGP+12] Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. Higher-order masking schemes for s-boxes. In Anne Canteaut, editor, *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 366–384. Springer, 2012.

[CGPZ16] Jean-Sebastien Coron, Aurelien Greuet, Emmanuel Prouff, and Rina Zeitoun. Faster evaluation of sboxes via common shares. Cryptology ePrint Archive, Report 2016/572, 2016. `http://eprint.iacr.org/`. Full version of this paper.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *CRYPTO*, 1999.

[Cor14] Jean-Sébastien Coron. Higher order masking of look-up tables. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 441–458, 2014.

[CPRR13] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 2013.

[CPRR15] Claude Carlet, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Algebraic decomposition for probing security. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 742–763. Springer, 2015.

[CRV14] Jean-Sébastien Coron, Arnab Roy, and Srinivas Vivek. Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. In *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, pages 170–187, 2014.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 423–440, 2014.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 850–867, 2012.

[GPS14] Vincent Grosso, Emmanuel Prouff, and François-Xavier Standaert. Efficient masked s-boxes processing - A step forward -. In David Pointcheval and Damien Vergnaud, editors, *Progress in Cryptology - AFRICACRYPT 2014 - 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings*, volume 8469 of *Lecture Notes in Computer Science*, pages 251–266. Springer, 2014.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 463–481, 2003.

[KJJ99] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In M.J. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397. SV, 1999.

[MS06] Stefan Mangard and Kai Schramm. Pinpointing the side-channel leakage of masked AES hardware implementations. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 76–90. Springer, 2006.

[NRS11] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *J. Cryptology*, 24(2):292–321, 2011.

[PR11] Emmanuel Prouff and Thomas Roche. Higher-order glitches free implementation of the aes using secure multi-party computation protocols. In *CHES*, pages 63–78, 2011.

[PR13] Emmanuel Prouff and Matthieu Rivain. Higher-Order Side Channel Security and Mask Refreshing. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013 - 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *LNCS*, pages 142–159. SV, 2013.

[RBN+15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, pages 413–427, 2010.

[RV13] Arnab Roy and Srinivas Vivek. Analysis and improvement of the generic higher-order masking scheme of fse 2012. In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES*, volume 8086 of *Lecture Notes in Computer Science*, pages 417–434. Springer, 2013.

[Sha79] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, November 1979.